Reti di Calcolatori T
Appello del 14/02/2021
Compito 2

Cognome: ……
Nome: ……
Matricola: ……

Tempo a disposizione: 3h

È obbligatorio inserire Cognome, Nome, e numero di Matricola all'inizio di ogni file sorgente, pena la non valutazione del compito, che viene stampato in modo automatico solo in caso siano presenti gli elementi detti sopra.
Si devono consegnare **tutti i file sorgente e tutti gli eseguibili prodotti singolarmente** (*per favore, solo quelli relativi ai file sorgente consegnati!!!*). Non consegnate zip, ma i file singoli.

La prova intende valutare le capacità progettuali e di programmazione sia in **ambiente Java** che in **ambiente C**, pertanto è consigliato sviluppare **entrambe** le soluzioni richieste al meglio.
In entrambi gli esercizi, sia in Java che in C, si effettuino gli opportuni controlli sui parametri della richiesta e si gestiscano le eccezioni, tenendo presente i criteri secondo cui si possa ripristinare il funzionamento del programma oppure si debba forzarne la terminazione.
Leggete con attenzione le specifiche del problema prima di impegnarvi "a testa bassa" nello sviluppo delle singole parti. Naturalmente, ci aspettiamo che i componenti da consegnare siano stati provati e siano funzionanti.

****

Si richiede il progetto della gestione dei servizi **RentAScooter**, per la gestione delle prenotazioni di monopattini. I servizi di **RentAScooter** mantengono, per ogni prenotazione, le seguenti informazioni: l'**id seriale del monopattino** (unico all'interno del sistema), la **carta di identità** dell'utente (una stringa di 5 interi), la **marca di monopattino prenotato** (una stringa che può assumere uno dei seguenti valori: **brand1**, **brand2, brand3**), le **immagini del monopattino**. I file con le immagini relative ad un monopattino sono mantenuti in una cartella denominata '*<seriale>_img*' situata nel direttorio corrente dove vengono lanciati server e client. In particolare, si richiede di realizzare le seguenti funzionalità:

1. **download delle foto del monopattino in base all'id seriale**: questa operazione richiede l'id seriale e scarica le immagini presenti nella cartella relativa al monopattino;
2. **aggiornamento della carta di identità**: nel caso in cui cambi il guidatore, questa operazione richiede l'id seriale e il numero di carta di identità del nuovo guidatore, quindi aggiorna la struttura dati;
3. **eliminazione di un monopattino**: questa operazione richiede l'id seriale, quindi controlla che non ci siano prenotazioni attuali e elimina i file immagine relativi, aggiornando il file system del server;
4. **visualizzazione di tutte le prenotazioni di una certa marca di monopattino** (brand1, brand2, brand3)**, messo in strada successivamente al 2011**: questa operazione richiede il brand e stampa a video la lista delle prenotazioni richieste. Si noti che gli id seriali vengono generati in successione cambiando le prime due lettere (es. **AA**xxxXX, **AB**xxxXX, **AC**xxxXX, …, **ED**xxxXX), consideriamo gli id seriali del 2011 quelli che iniziano con le lettere '*ED'*.

Si progetti con particolare attenzione la **struttura dati** che mantiene lo stato, fino ad un massimo di N prenotazioni (L, per libero a default), da implementare opportunamente nei diversi ambienti richiesti, Java e C.

Id seriale monopattino	Carta di identità	Brand del monopattino	Folder immagini
AN745NL	**00003**	**brand1**	**AN745NL_img/**
FE457GF	**L**	**brand2**	**FE457GF_img/**
L	**L**	**L**	**L**
…	**…**	**…**	**…**
NU547PL	**40063**	**brand1**	**NU547PL_img/**
LR897AH	**56832**	**brand2**	**LR897AH_img/**
MD506DW	**00100**	**brand3**	**MD506DW_img/**

Si considerino e si segnalino le possibilità di interferenze fra le operazioni, evitandole dove necessario.

Parte Java

Sviluppare un'applicazione C/S basata su **socket stream** che realizzi le operazioni remote per:

- **aggiornare il numero di carta di identità**;
- **download delle foto di un monopattino richiesto:**

utilizzando **un'unica connessione per ogni sessione cliente**: questo vincolo è da intendersi come **prioritario e fondamentale**.
Più in dettaglio:

- Il **cliente** è organizzato come un **processo filtro ciclico che consuma l'input fino a fine file** e, per ogni iterazione del ciclo, chiede all'utente quale tipo di operazione vuole effettuare e realizza le interazioni col server utilizzando **una sola connessione per la intera sessione**; alla ricezione di fine file, libera opportunamente le risorse e termina.
  Per ogni richiesta ricevuta dall'utente, il client prima invia il tipo di servizio al server, poi gestisce gli invii e le ricezioni necessarie alla realizzazione dello specifico servizio richiesto.
  Nel caso della funzionalità di **aggiornamento della carta di identità**, per ogni richiesta, il client chiede all'utente e invia al server l'id seriale e il nuovo numero di carta di identità; quindi riceve l'esito dell'operazione stampandolo a video.
  Nel caso di **download delle foto**, per ogni richiesta, il client chiede all'utente l'id del monopattino ed esegue la multiple-get delle foto relative al monopattino, salvandole in locale.
- Il **server** è organizzato come un **unico processo che gestisce in modo parallelo** l'interazione coi clienti, generando un figlio per tutta la sessione di richieste da quel client. Per ogni richiesta, il processo figlio che serve la sessione con una prima lettura discrimina il tipo di funzionalità richiesto, poi gestisce opportunamente l'operazione e si pone in attesa di nuove richieste dallo stesso client; alla lettura della fine sessione, il figlio termina.
  Nel caso della funzionalità di **aggiornamento della carta di identità,** il figlio riceve l'id seriale e il nuovo numero di carta di identità, quindi aggiorna la struttura dati e restituisce un intero positivo che indica l'esito dell'operazione: 0 in caso di successo, -1 in caso di problemi.
  Nel caso di **download delle foto**, il figlio riceve l'id del monopattino e quindi invia nome e contenuto di tutti i file delle immagini richieste usando l'unica connessione della sessione.

Parte C

Utilizzando RPC sviluppare un'applicazione C/S che consenta di effettuare le **operazioni remote** per:

- **visualizzare tutte le prenotazioni di una certa marca di monopattino (brand1, brand2, brand3), messo in strada successivamente al 2011;**
- **eliminare un monopattino.**

Il progetto prevede che il server metta a disposizione due procedure (parte di una interfaccia specificata in XDR e contenuta nel file *RPC_xFile.x*) invocabili in remoto dal client:

- la procedura **visualizza_prenotazioni** accetta come parametro d'ingresso **il brand del monopattino**, e restituisce la lista (array) delle prenotazioni di monopattini del brand richiesto, la cui messa in strada sia avvenuta successivamente all'anno 2011, cioè aventi come prime due lettere dell'id seriale un valore maggiore di '*ED*'. Restituire al massimo i primi 6 risultati utili.
- la procedura **elimina_monopattino** accetta come parametro d'ingresso **l'id seriale di un monopattino**, controlla che non ci siano prenotazioni attuali, elimina tutti i file immagine relativi e restituisce un intero con l'esito dell'operazione.

Si progettino inoltre i sorgenti:

- **RPC_Server** (contenuta nel file *RPC_Server.c*), che implementa le procedure del server invocabili in remoto;
- **RPC_Client** (contenuta nel file *RPC_Client.c*), il processo filtro che realizza l'interazione con l'utente, propone ciclicamente i servizi che utilizzano le due procedure remote, e stampa a video i risultati, fino alla fine dello stream di input.